ARMY RESEARCH LABORATORY

BuyIt Detailed Design Report: A C-BASS Component

by Brian R. Schallhorn

ARL-MR-451 July 1999

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5066

ARL-MR-451 **July 1999**

BuyIt Detailed Design Report: A C-BASS Component

Brian R. Schallhorn
Corporate Information and Computing Center, ARL

Abstract

This document is the third in a series of reports resulting from a structured engineering process for the BuyIt Software Project. As part of the Corporate Business Application Software System (C-BASS) suite of work flow applications, BuyIt automates small purchase orders for the U.S. Army Research Laboratory (ARL). Three major sections of this report give background for the BuyIt Project, explain the structured design and methodology, and define the design environment. A fourth segment presents a detailed design for BuyIt using the formalisms of structure charts for defining operations and module specifications for defining objects.

Table of Contents

List of Figures

INTENTIONALLY LEFT BLANK.

List of Tables

INTENTIONALLY LEFT BLANK.

1. Introduction

As the demand for cost-effective research and development (R&D) capabilities increases, the U.S. Army Research Laboratory (ARL) looks to modern information technology (IT) as an enabler for core organizational practices. As part of a general trend toward streamlining and automating critical transactions, the Corporate Business Application Software System (C-BASS) is a suite of software applications characterized by

- Robustness and scalability,
- Maximization of COTS,
- Compatibility with standard systems,
- Focus on interoperability and elimination of "stovepipes," and
- Emphasis on security.

1.1 BuyIt Prototype. As a component of C-BASS, the purpose of the BuyIt project is to model a secure client/server system that processes small purchase requests (under $100,000). The motivating force behind this project has been ARL downsizing and the findings put forth in the Business Process Reengineering (BPR) report [1] and the BPR "To-Be Model: Small Purchase." This model identifies potential process improvements (some of which require computer automation) that would increase productivity of purchasing operations at ARL.

The proof-of-principle BuyIt prototype will alleviate some of the risks involved in implementing new technologies used to build the emerging ARL Intranet. As currently envisioned, Intranet capabilities will consist of an integrated set of Lotus Notes and Web-based software to support R&D competencies. Such a large and complex undertaking requires both careful planning and proactive management. Additionally, the BuyIt project will help to refine requirements described in the ARL BPR "To-Be Model" [1]. Development of a full production BuyIt will proceed in phases, using an incremental, evolutionary approach that emphasizes integration of "lessons learned" from the prototype design, implementation, and evaluation.

1.2 Antecedent Documents and Contents of This Report. Two documents precede this report: "BuyIt Software Development Plan" [2] and "BuyIt Software Requirements Analysis" [3]. The first gives an overview of the system's purpose, identifies organizational requirements and constraints, develops a management plan, and sets forth a timetable with major milestones. The second document elaborates on the BuyIt design through the formalisms of data flow diagrams, structured English narratives of the system's functionality, and a data dictionary.

This report contains four sections:

- "Structured Design Methodology - briefly explains the methodology used to extract the software functional specifications.
- "High-Level Requirements and Design Overview" - describes the design opportunities and constraints, identifies a software reuse strategy, and presents the basic features of the Lotus Notes development environment.
- "Operations Overview" - presents operational scenarios, including user interfaces and interaction with legacy systems as well as core components of Lotus Notes.
- "Procedural Design Description" - breaks the system design into finer detail and presents structure charts and module specifications.

2. Structured Design Methodology

Modern software engineering relies on structured analysis to design coherent, reliable, and maintainable data-processing systems [4]. Using a process of step-wise refinement, the designer starts with a top-level, conceptual architecture for the system and moves downward, incrementally decomposing each component and providing greater specificity for each functional unit. The purpose of a structured design approach is to generate a detailed description of how to merge software objects that behave in agreement with the structured system analysis models and meet all other system requirements.

Usually coming last in an application's documentation library, the detailed design document gives functional specifications at the granularity required for a programming language implementation of each component. In other words, at this level of decomposition, the detailed design (also called the procedural design) defines each operation and each object within the system.

Within the software engineering process, a detailed design serves three purposes. First, the design is a product review, used to uncover logic errors prior to programming investment. Second, a codified detailed design serves as a template to guide coding. Third, the document helps with the derivation of test cases for software validation.

As the procedural design document for the BuyIt prototype, this report finalizes the design of all data structures internal to each component and provides a representation of program logic at a relatively low level of abstraction. Two important representations used in this document are structure charts and module specifications. The structure chart shows the division of a system into modules, displaying their hierarchical relationship, organization, connections, and interaction with one another. The module specifications give detailed information on how modules process the interface inputs in order to produce the necessary outputs. A BuyIt representation using these formalisms is given in section 5, "Design Description."

The following section reviews the design opportunities and constraints that resulted in the conceptual model of BuyIt. Section 4, "Operations Overview," takes a transactional approach to describing the system. First, BuyIt is viewed from the perspective of the end user by summarizing screen design and human-machine interaction. Then BuyIt's transactions with two legacy systems (Standard Operations and Maintenance Army Research Development System [SOMARDS] and Standard Army Automated Contract System [SAACONS]) are delineated. The last portion of section 4 covers BuyIt's transactions with core capabilities of Lotus Notes.

3. High-Level Requirements and Design Overview

Requirements for the C-BASS suite and its components accrued from three sources: (1) general organizational streamlining and BPR, (2) user needs conjoined with "best" practices, and (3) the opportunities and the limitations provided by existing hardware and software. Of special interest are both the features offered by Lotus Notes for groupware development and the issues stemming from legacy systems.

3.1 Lotus Notes Development Environment. Drawing momentum from the climate of organizational change, the Corporate Information and Computing Center (CICC) has designated computer-supported collaborative work (CSCW) applications as a cornerstone in ARL's emerging Intranet. As determined by an ARL task force, critical features for a viable Intranet computer architecture include

- Uniformity of communication among and within ARL sites,
- Use of Commerical Off-the-Shelf (COTS) products,
- Adherence to standards (American National Standards Institute [ANSI], Department of Defense [DOD], and de facto) wherever possible,
- Multiplatform support,
- Adequate security,
- Support for workflow operations to improve business processing, and
- Modern data management capabilities.

Lotus Notes was selected as a CSCW enabling technology because the product provides three invaluable capabilities:

- Potential for systematic reuse of applications built in LotusScript,
- A commercially viable, large-scale product with enough market share to ensure continued evolution,
- A complete development environment that provides generic/adaptable core modules for standard business practices.

Lotus Notes core components include a uniform user interface, primary data store capabilities, and workflow routing through a robust messaging engine. Additionally, the developer's library of generic applications includes a simple approval/routing example. Reuse of this system's basic logic reduces design and implementation effort, requiring only minimal expansion effort in the number of approval cycles to handle the approval and routing of BuyIt purchase requests.

Through the Lotus Notes "control agent" feature (similar to a "macro" in other software), developers can customize routine functionality to create powerful tools that reflect preferences in handling work. The "control agent" captures an event (or set of events) within the operation and triggers a subsequent action. For example, an agent may sort incoming email and place it into appropriate folders after being trigged by specified keywords in the subject field. Other agents may tailor database queries or filter and route results. Still others may look for specified events and then trigger a response, such as an automated reminder sent to an email listserv.

3.2 Legacy Efforts. During the BPR development effort, the former Enterprise Systems Branch (ESB) fielded an on-line interface to SOMARDS. The interface pulled purchase request funding data from an Oracle database and then used Expect (a software package automating interactive processes) to connect to the SOMARDS user interface and certify funds on-line. Reuse of this interface package requires only minor changes for the BuyIt prototype.

3.3 Major System Components. As illustrated in Figure 1, at a high level of representation, BuyIt contains four major components:

- User Interface - consisting of a set of input/output screens, this process allows requesters to enter data, functional users to approve requests, and all users to perform status inquiries and generate reports. The user interface is implemented using the Lotus Notes client, located on the user's desktop workstation.
- Data Store "PURCHASE REQUESTS" - implemented on the central Lotus Notes server, this entity contains all the active, canceled, and closed purchase requests that are or have been processed by BuyIt.

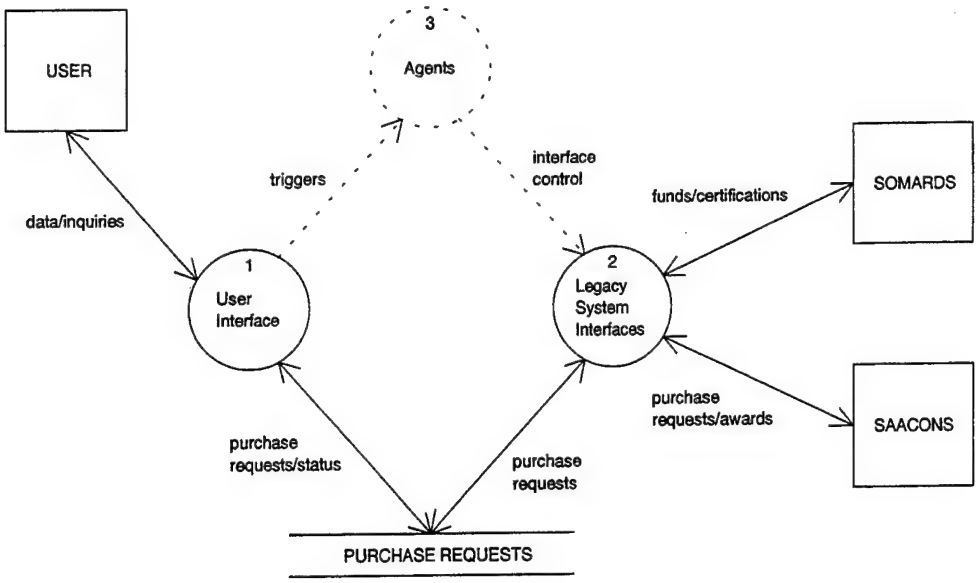

Figure 1. Overview of BuyIt Prototype System Design.

- Legacy System Interfaces - this process connects to the SOMARDS and SAACONS systems for fund certification and for instantiating purchase orders. Portions of the legacy system interfaces will be implemented on a Unix server and an Oracle database, with connects to the Notes server through the network.
- Control Agents - analogous to an event-oriented application, this process receives signals (called "triggers") from the user interface, which in turn controls the timing of the legacy system interactions.

4. Operations Overview

Besides showing the overall design of BuyIt, Figure 1 also helps identify the system interaction scenarios. Representing these scenarios are graphics of the user interface (the screens options described in following text) and the background processes that communicate with the legacy system.

4.1 Human-Machine Interaction. Modern design of user interfaces includes highly visual menu options for selecting processes, iconic representations for system features, and forms

displays for data input/output and information consolidation. All major functionality for BuyIt users exists within five central interface screens: (1) Startup, (2) Initiate Purchase Request, (3) Approval, (4) Status, and (5) Cycle Time Report. Figures 2–6 more fully illustrate human-machine interactions indicated on each screen.

4.1.1 Startup Screen. When a user first logs into BuyIt, the startup (or inbox) screen shown in Figure 2 appears. The large area on the right is the user's inbox, where requests requiring attention are listed. The inbox contents are based on the individual user's role. When the user selects a particular request, the system displays the entire purchase request and allows the user to perform processing (e.g., approval, data entry, etc.).

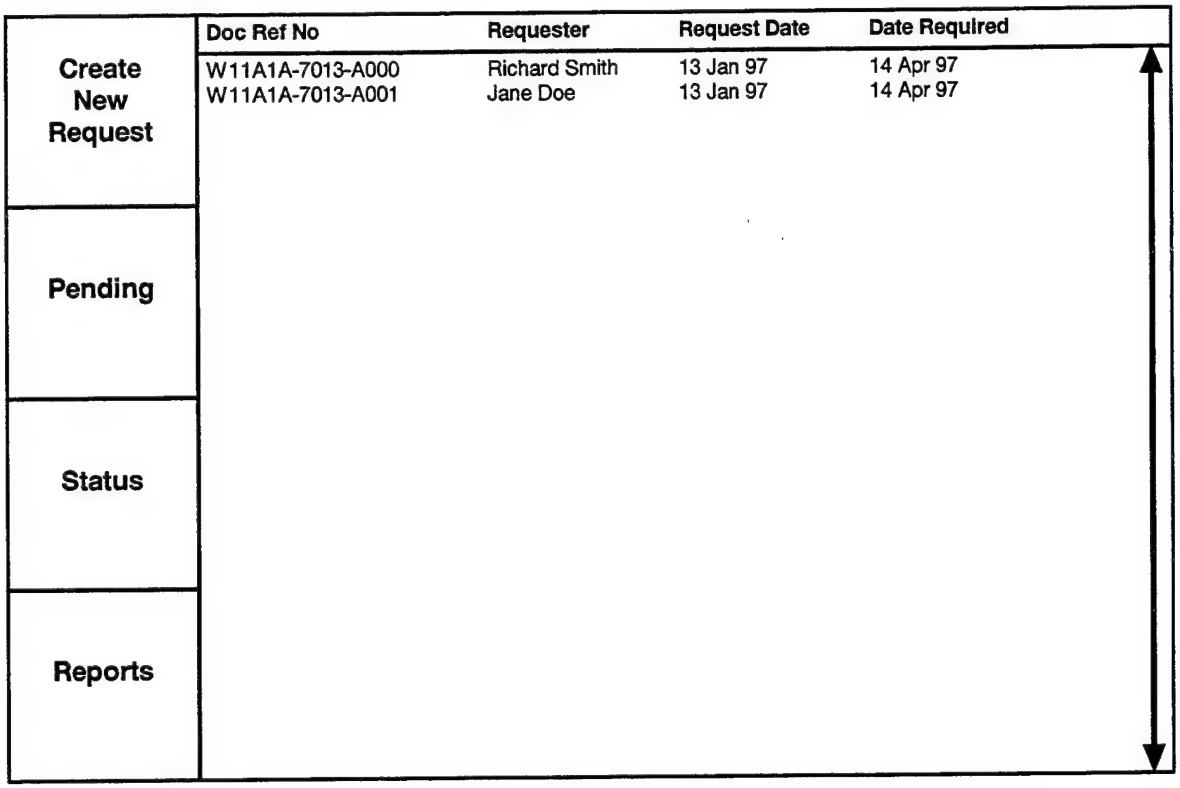

Figure 2. BuyIt Startup/Inbox User Screen.

Along the left-hand side of the inbox screen are buttons by which the user navigates to the various functional areas of BuyIt. Four functions are indicated:

- Create New Request - initiates a new purchase request; fills in the requester information; displays the purchase request entry screen.
- Pending - displays or refreshes the inbox screen.
- Status - determines the current status of purchase requests that have been "touched" by the user and displays results on the status inquiry screen.
- Reports - generates the cycle time report and displays it on the reports screen.

4.1.2 Purchase Request Entry Screen. Figure 3 illustrates the "Purchase Request Entry" screen. Based on the userid of the requester, the form will be filled in with the requester's name, office symbol, phone, and address. The requester then fills in the rest of the information (e.g., date required, priority, vendors, items, fund source, etc.). By pressing the appropriate button along the top of the screen, the requester can

- cancel the request,
- save the request for later data entry, or
- submit the request to their supervisor for approval.

4.1.3 Approval Screen. Figure 4 shows the "Approval" screen. This display appears when the user selects a request from his or her inbox. Full purchase details are displayed for the approving official. Depending on his or her role, the official may have editing privileges for some of the fields. Privileges are assigned as follows:

- supervisor - full editing rights on all fields,
- budget analyst - editing of fund source,
- property book officer - item tags,
- contracting officer - buyer assignment, and
- receiving - item tags.

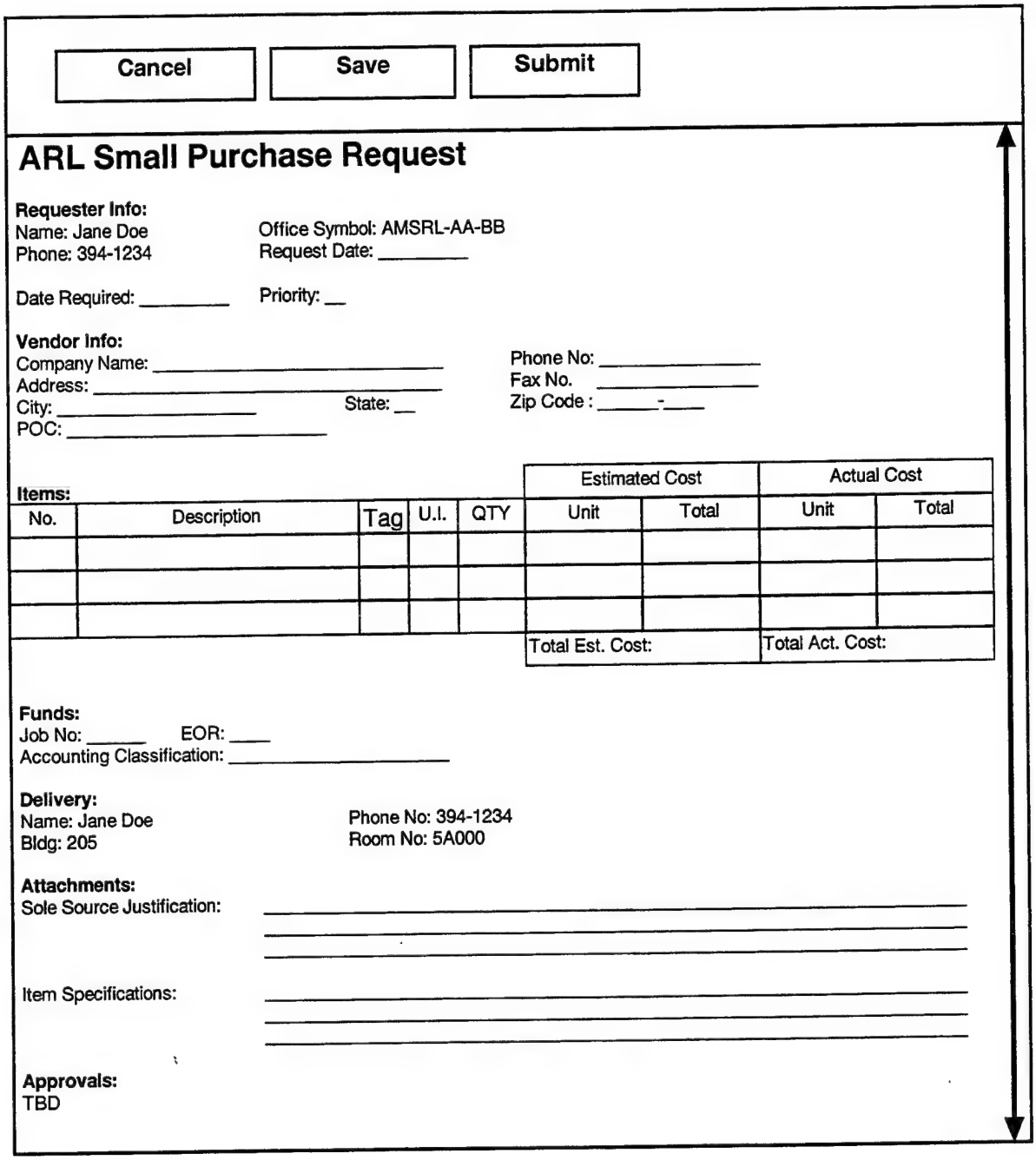

Figure 3. Purchase Request Entry Screen.

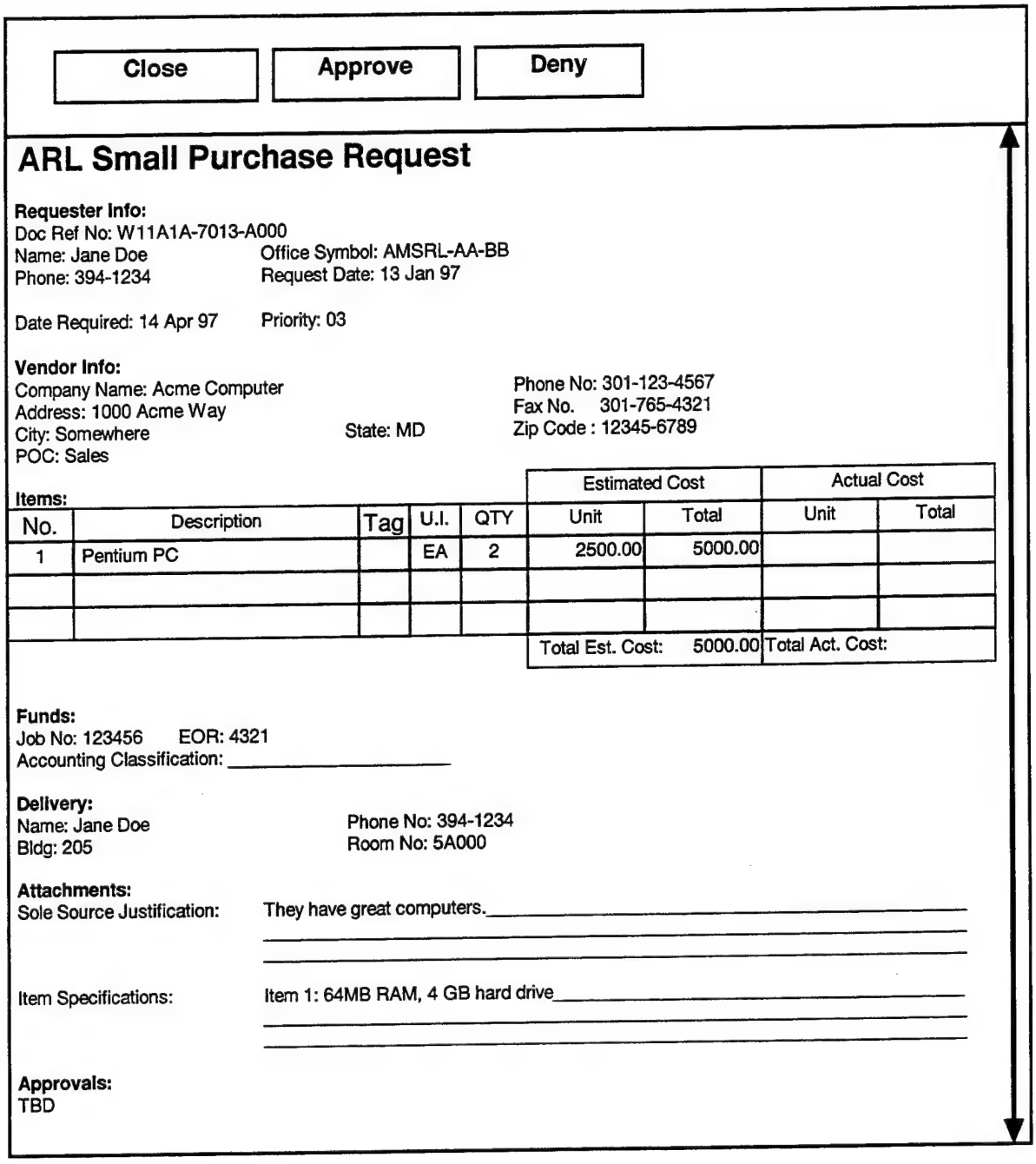

Figure 4. Approval Screen.

Once the user has reviewed the request, he or she can press the appropriate button to

- close the display and save approval for later,
- approve the request, or
- deny approval and enter an explanation.

4.1.4 Status Inquiry Screen. The "Status Inquiry" screen is shown in Figure 5. This interface displays the requests that have been "touched" by the user, either as requester or approving official. This screen also indicates the current status. By selecting a request, the full details are displayed as shown in the "Approval" screen (Figure 4), but without the "Approve" and "Deny" buttons. Additionally, editing of the fields is not available at this point.

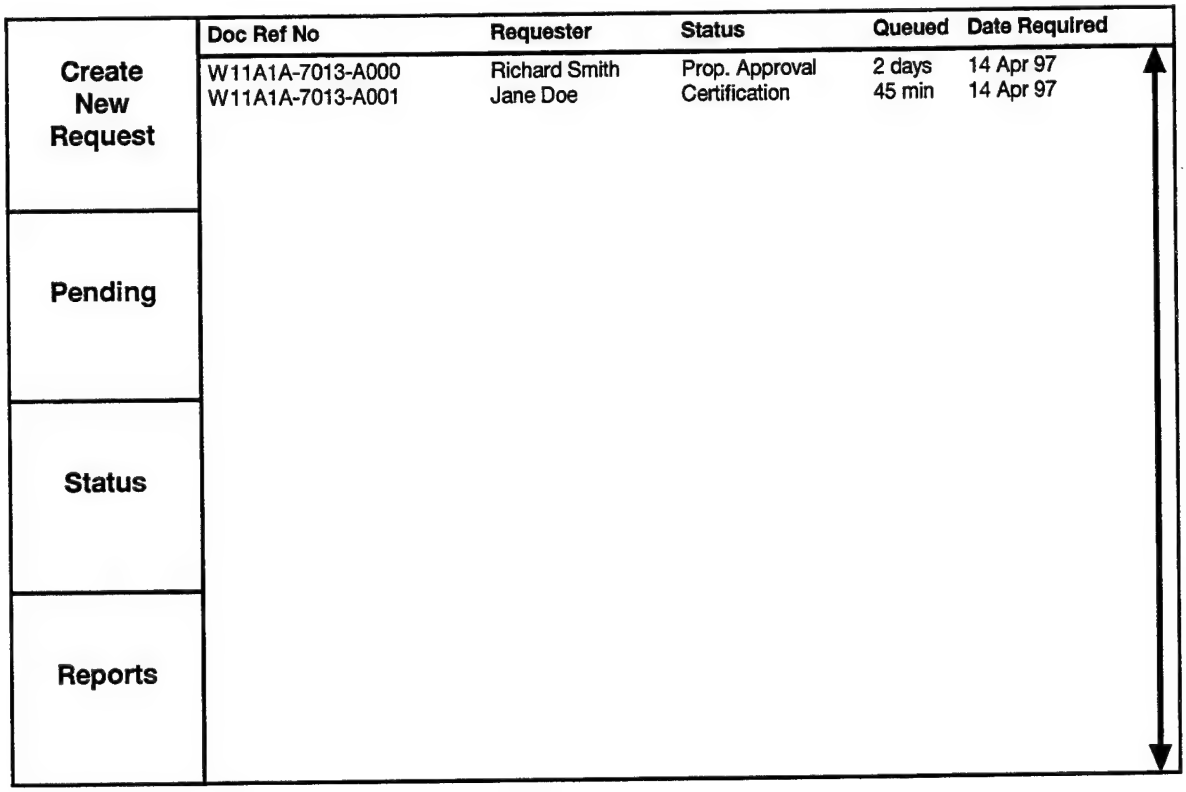

Figure 5. Status Inquiry Screen.

4.1.5 Reports Screen. Figure 6 depicts the "Reports" screen. This screen displays the "Cycle Time Report," which shows the average time a request waits for approval in the various functional areas. As BuyIt matures and additional reports become available, users can select from this enriched set of features by using the navigation buttons along the left side of the screen.

4.2 Software-to-Software Interactions. In addition to the user-initiated interactions, BuyIt's background processes communicate both with legacy systems (SOMARDS and SAACONS) and with the groupware functions provided in the Lotus Notes development environment. Each is more fully described later.

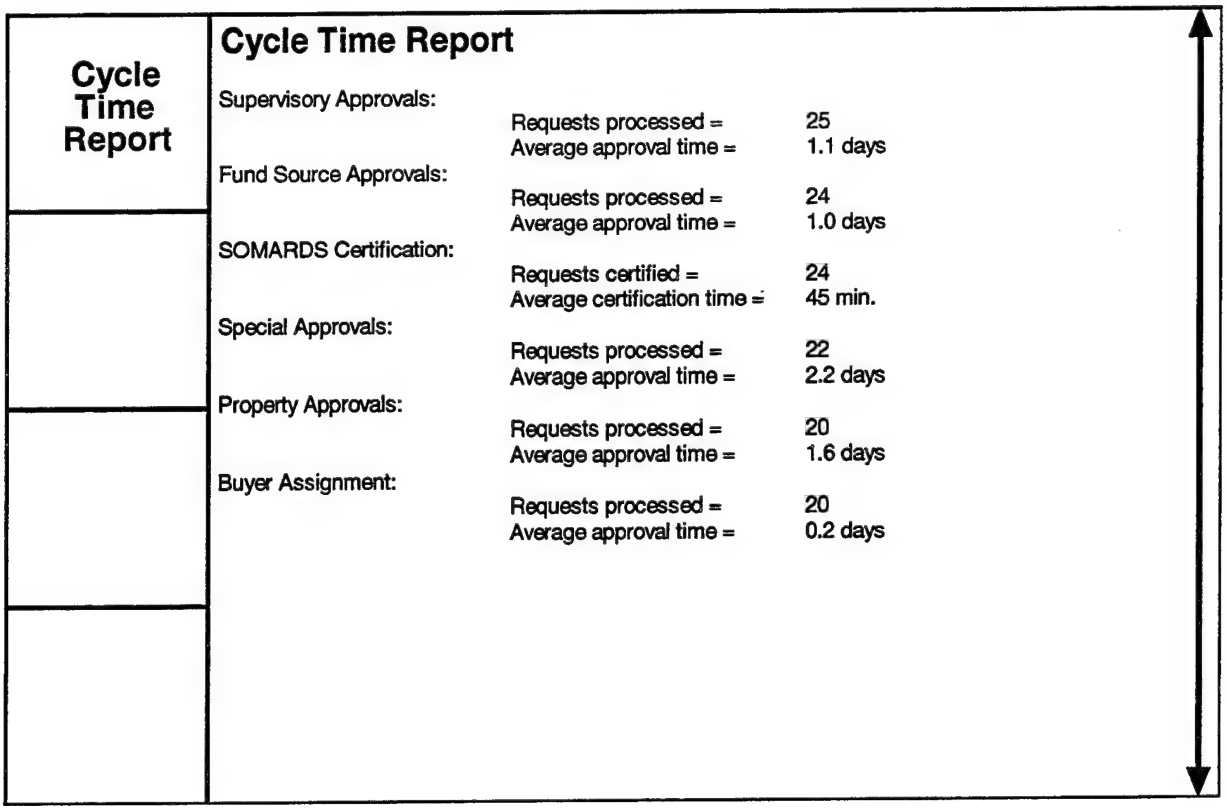

Figure 6. Reports Screen.

4.2.1 SOMARDS. The SOMARDS interface operates in three modes: (1) build a transaction block, (2) certify funds, and (3) reconcile the transaction block. Each operation uses the SOMARDS on-line user interface. Building a block occurs first thing in the morning, after SOMARDS is brought up. The block is named and the initial balance set to zero. This transaction block is then used during the course of the day for all certifications processed by BuyIt. As they are submitted, purchase requests are deemed eligible for certification and then are placed in a first-in-first-out (FIFO) queue. The certification process continually checks the queue and processes the "certifiable requests" as they become available. Depending on the number of pending transactions, the response time could be a few seconds or a few minutes.

Certification results are placed in a second queue, which is checked by a Lotus Notes control agent. This processing element routes certified purchase requests to their next destination, while rejected purchase requests are sent back to the requester for indicated corrections. Reconciliation occurs just before SOMARDS is brought down at the end of the day. The cumulative total is read from SOMARDS and checked against a running total calculated from BuyIt. Any discrepancies are sent to the BuyIt administrator, and the block is reconciled using the SOMARDS cumulative value.

4.2.2 SAACONS. The SAACONS interface operates in two modes: (1) uploading purchase request data to SAACONS and (2) downloading award data from SAACONS.

The upload occurs after the contracting officer assigns a buyer to the request (refer to "BuyIt Software Requirements Analysis" [3] for process/data flow). The purchase request details are transferred to an upload queue residing in an Oracle database. At particular intervals, the queue is dumped to a file, formatted for SAACONS, and transferred to the SAACONS host. The reverse of this process accomplishes the SAACONS download of award data.

At prescribed time intervals, SAACONS award data are transferred to the BuyIt system and loaded into a temporary Oracle database queue. At this point, a Notes control agent scans the queue and updates any purchase requests with new award data.

4.2.3 Lotus Notes Control Agents. The control agents reside on the Notes server and operate in two modes: either triggered by the user interface client or automatically executed at prescribed intervals.

Triggers from the user interface consist of (1) fund source or actual cost approvals (SOMARDS certification) or (2) a buyer assignment (SAACONS upload). The funding approval event activates a Notes control agent, which puts the information required by SOMARDS into a FIFO queue. The buyer assignment action triggers a different Notes control agent, which puts the purchase request information into a separate FIFO queue. Both queues reside in an Oracle database and are processed as described previously.

Using server-based control agents achieves a significant economy. If the user interface alone were used to communicate with the queues, the Oracle database network connection software (SQL*Net) would be required for each client. Thus, locating the agents on the server and triggering them from the user interface saves the cost and administration of having the extra client software.

In addition to being event-driven, these Notes server agents also execute at prescribed intervals. Whether event triggered or interval initiated, the control agents process information received from the legacy systems, as illustrated in the next two examples.

The SOMARDS fund certification agent interprets the message returned from SOMARDS and marks the request either as certified or rejected. If certified, the request is routed to the next destination. If rejected, an error message is placed in the "explanation" field and the request is returned to its originator.

The SAACONS award download agent checks the total actual cost against the certified total estimated cost and fills in the purchase order information once the award has been made. If the total actual cost is greater than the total estimated cost, then the request is sent to the supervisor for "actual cost" approval. Once the status of all items for a particular request has been set to

"awarded," the agent fills in the purchase request with the purchase order information (purchase order number, selected vendor, actual cost, and delivery date).

5. Procedural Design Description

At the procedure level of a structured design document, the information system is partitioned into modules and transactions among these elements are defined. Within this section, two formalisms are used to give a detailed design specification for BuyIt. First, structure charts provide a graphic representation of the modules and their hierarchical structure. These charts also show the data/information transactions and central interfaces among the modules. Second, specifications for each procedural component (module) are given using four categories of definition: purpose, uses, returns, and functional details.

Table 1 is a composite of BuyIt, indicating twelve functional components, a decomposition of each into its associated modules, and the figure number for the structure chart that shows each functional component as a cluster of modules and interactions.

5.1 System Navigation. The major components of BuyIt are shown in the structure chart displayed in Figure 7. The "Navigation" function represents the user interface, and the "LI" and "AG" continuation markers represent—respectively— the legacy system interfaces and the Notes control agents (which are shown in more detail in later segments). The continuation markers "PPR" (Prepare Purchase Request), "PPPR" (Process Pending Purchance Request), "SI" (Status Inquiry), and "GR" (Generate Report) point to the structure charts that represent the other functional areas of the user interface.

Specifications for the five modules appearing in Figure 7 are listed as follows.

(1) **Module:** BuyIt.

Purpose: Provide a user interface for purchase request processing and an automated interface to legacy systems.

Table 1. Overview of Structure Charts and Module Specifications

Functional Component	Associated Module Specifications	Structure Chart
Navigation	- BuyIT - Navigation - Show Inbox - Get Pending Purchase Requests - Select Action	Figure 7
Prepare Purchase Request	- Prepare Purchase Request - Create new Purchase Request - Fill in Purchase Request - Fill in Fund Source	Figure 8
Process Pending Purchase Request	- Process Pending Purchase Request - Cancel Purchase Request - Approve Shipment	Figure 9
Obtain Approvals	- Obtain Approvals - Purchase Request - Approve Fund Source - Approve Actual Cost - Queue Funding - Approve Special Items - Approve Property - Approve Purchase	Figure 10
Edit Purchase Request	- Edit Purchase Request - Edit Rejected Purchase Request - Attach Item Tags - Assign Buyer - Queue Assigned Request	Figure 11
Receive Shipment	- Receive Shipment - Receive Order - Tag Shipment Items	Figure 12
Status Inquiry	- Status Inquiry	Figure 13
Generate Report	- Generate Report	Figure 14
SOMARDS Legacy Interfaces	- SOMARDS Legacy Interfaces - Connect to SOMARDS - Build Block - Send Block Data - Get Build Return Message - Reconcile Block - Send Reconcile Info - Get Reconcile Return Message - Disconnect from SOMARDS	Figure 15

Table 1. Overview of Structure Charts and Module Specifications (continued)

Functional Component	Associated Module Specifications	Structure Chart
Certify Funds	- Certify Funds - Get Next Entry from Queue - Send Funds Info - Get Return Message - Update Daily Balance	Figure 16
SAACONS Interface	- SAACONS Upload - Get Next Upload from Queue - Send Purchase Request Details - SAACONS Download - Receive Purchase Order Details - Update Purchase Request	Figure 17
Lotus Notes Agent Processes	- Agents - Certification Queue Agent - SOMARDS Return Agent - SAACONS Upload Agent - SAACONS Award Agent	Figure 18

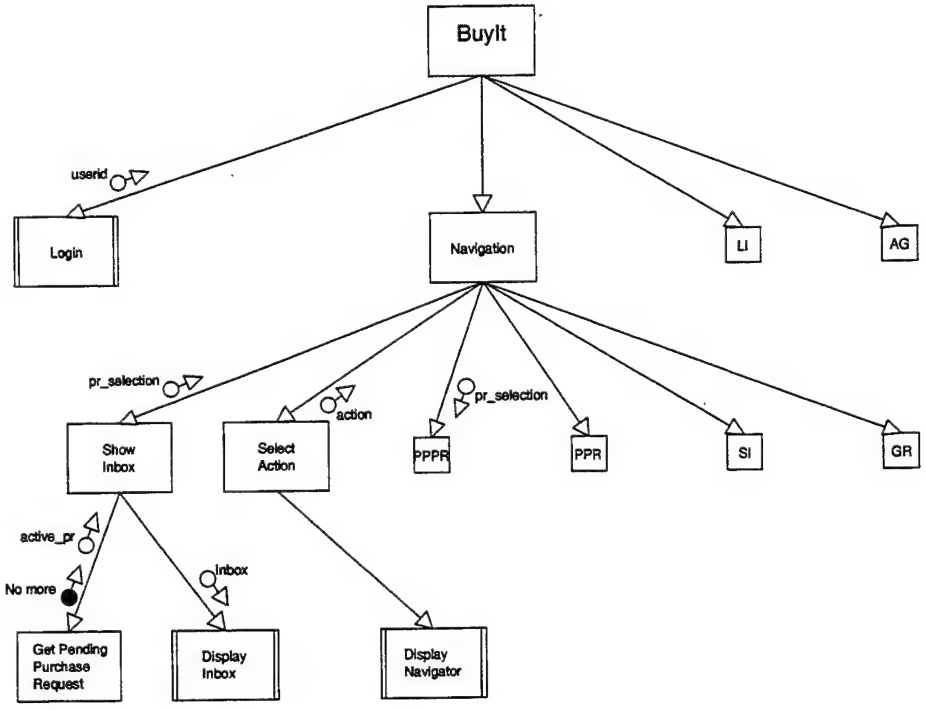

Figure 7. BuyIt Major Components and User Interface Navigation.

Uses: Login, Navigation, Legacy Interfaces.

Returns: N/A.

Functional Details:

1. Initiate Agents.
2. Initiate Legacy Interfaces.
3. Wait for open database attempt and get Login.
4. If login is successful, start Navigation user interface.

(2) **Module:** Navigation.

Purpose: Allow user access to the various functional areas of the user interface.

Uses: Show Inbox, Select Action, Prepare Purchase Request, Process Pending Purchase Request, Status Inquiry, Generate Report.

Returns: N/A.

Functional Details:

1. Wait for Show Inbox or Select Action to return pr_selection or action.
2. If pr_selection is not null, then Process Pending Purchase Request using pr_selection.
3. If action is not null, then, depending on action selected, Prepare Purchase Request or Status Inquiry or Generate Report.

(3) **Module:** Show Inbox.

Purpose: Display to user purchase requests requiring their attention and allow user to select a particular request for processing.

Uses: Get Pending Purchase Requests, Display Inbox.

Returns: pr_selection.

Functional Details:

1. Get Pending Purchase Requests until no more pending requests.
2. Display Inbox using pending requests.
3. Wait for user to select a particular request.
4. Return pr_selection to parent module.

(4) **Module:** Get Pending Purchase Requests.

Purpose: Retrieve purchase requests requiring the user's attention.

Uses: N/A.

Returns: active_pr.

Functional Details:

1. Scan the ACTIVE data store for requests with user or role set as the author.
2. Return the active_pr or control signal if no more.

(5) **Module:** Select Action.

Purpose: Allow user to select a particular functional area of the user interface.

Uses: Display Navigator.

Returns: action.

Functional Details:

1. Display Navigator.
2. Wait for user to select action using navigator buttons.
3. Return selected action to parent module.

5.2 Prepare Purchase Request. Figure 8 is the structure chart for the "Prepare Purchase Request" process. This function creates a new purchase request and allows the requester to fill in necessary data elements.

Specifications for the four modules associated with Figure 8 are listed as follows.

(1) **Module:** Prepare Purchase Request.

Purpose: Allow requester to fill in a purchase request and submit it for approval.

Uses: Create New Purchase Request, Display Blank Purchase Request, Fill in Purchase Request, Fill in Fund Source.

Returns: funded_pr.

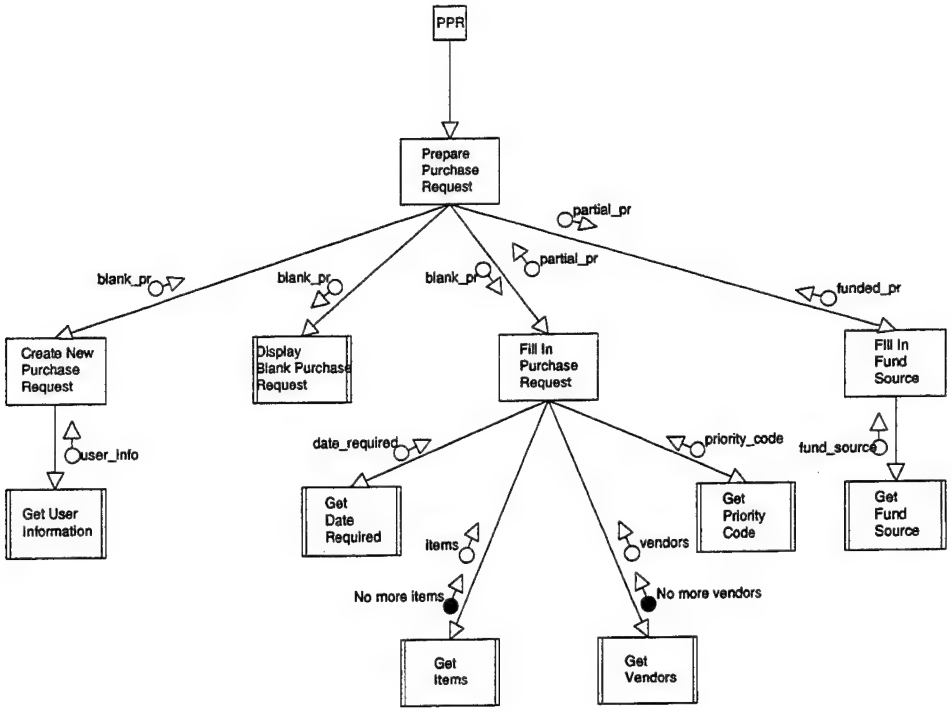

Figure 8. Prepare Purchase Request.

Functional Details:

1. Create New Purchase Request using the userid.
2. Display Blank Purchase Request for user input.
3. Fill in Purchase Request and Fill in Fund Source.
4. Wait for user to submit, save, or cancel new purchase request.
5. If cancel, go to inbox screen.
6. If save, store purchase request in ACTIVE.
7. If submit, determine supervisor userid, set request author to supervisor.

(2) **Module:** Create New Purchase Request.

Purpose: Create a new purchase request with the requester's user_info filled in.

Uses: Get User Information.

Returns: blank_pr.

Functional Details:

1. Get User Information using requesters userid.
2. Set the requester user_info in the new purchase request.
3. Return the blank_pr to parent module.

(3) **Module:** Fill in Purchase Request.

Purpose: Fill in the blank purchase request.

Uses: Get Date Required, Get Items, Get Vendors, Get Priority Code.

Returns: partial_pr.

Functional Details:

1. Get Date Required and set in the purchase request.
2. Get Items and set in the purchase request until no more items.
3. Get Vendors and set in the purchase request until no more vendors.
4. Get Priority Code and set in the purchase request.
5. Return partial_pr to parent module.

(4) **Module:** Fill in Fund Source.

Purpose: Fill in the fund source information.

Uses: Get Fund Source.

Returns: funded_pr.

Functional Details:

1. Get Fund Source and set in the purchase request.
2. Return funded_pr to parent module.

5.3 Processing Pending Purchase Request. Figure 9 shows the "Process Pending Purchase Request" structure chart. This functional component allows users of BuyIt to process purchase requests, an event involving several actions (e.g., approvals, edit, cancel, receive, and accept shipment).

Specifications for the three modules associated with Figure 9 are listed as follows.

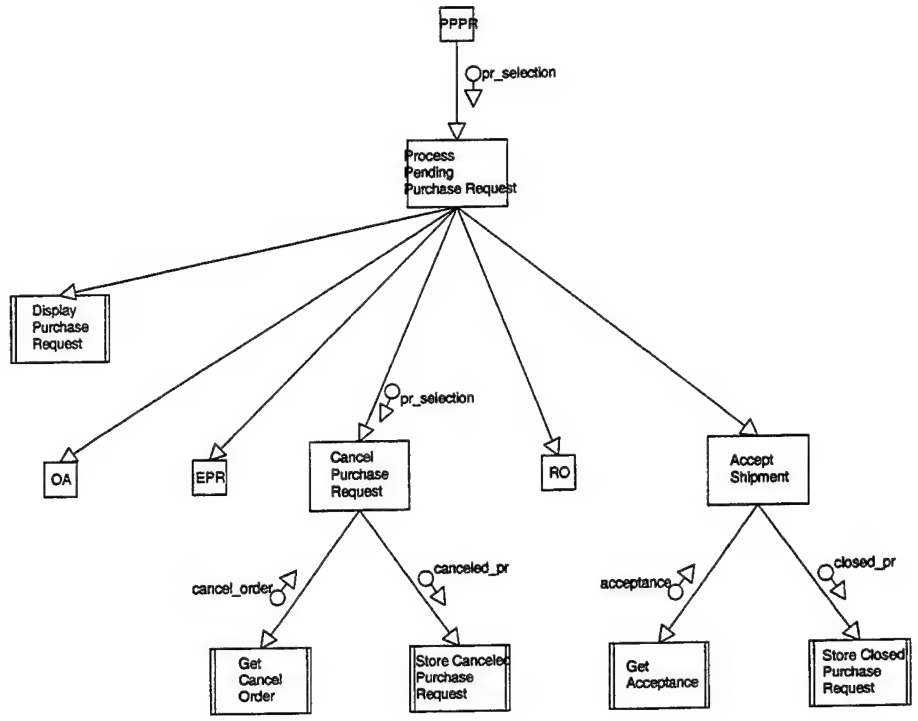

Figure 9. Process Pending Purchase Request.

(1) **Module:** Process Pending Purchase Request.

Purpose: Provide a means for users to process pending purchase requests.

Uses: Display Purchase Request, Obtain Approvals, Edit Purchase Request, Cancel Purchase Request, Receive Order, Accept Shipment.

Returns: N/A.

Functional Details:

1. Display Purchase Request to user.
2. If user role matches the purchase request next approval, then Obtain Approvals.
3. If user is the requester and pr_selection is a rejected_pr, then Edit Purchase Request.
4. If user role is property and pr_selection is a special_pr, then Edit Purchase Request.
5. If user role is contracting officer and pr_selection is a orderable_p, then Edit Purchase Request.

6. If user is the requester or requester's supervisor, then allow Cancel Purchase Request.
7. If user role matches receiving and pr_selection is a ordered_pr, then Receive Order.
8. If user is the requester and pr_selection is a tagged_pr, then Accept Shipment.

(2) **Module:** Cancel Purchase Request.

Purpose: Allow the requester or supervisor to cancel the request.

Uses: Get Cancel Order, Store Canceled Purchase Request.

Returns: N/A.

Functional Details:

1. Get Cancel Order from user.
2. If cancel order is "Yes," then Store Canceled Purchase Request in the CANCELED data sore.

(3) **Module:** Approve Shipment.

Purpose: Allow the requester to accept or reject shipment items.

Uses: Get Acceptance, Store Closed Purchase Request.

Returns: N/A.

Functional Details:

1. Get Acceptance from the requester.
2. Store Closed Purchase Request in CLOSED data store.

5.4 Obtain Approvals. Figure 10 shows the "Obtain Approvals" process structure chart.

Specifications for the eight modules associated with Figure 10 are listed as follows.

(1) **Module:** Obtain Approvals.

Purpose: Obtain approvals from the necessary approving officials.

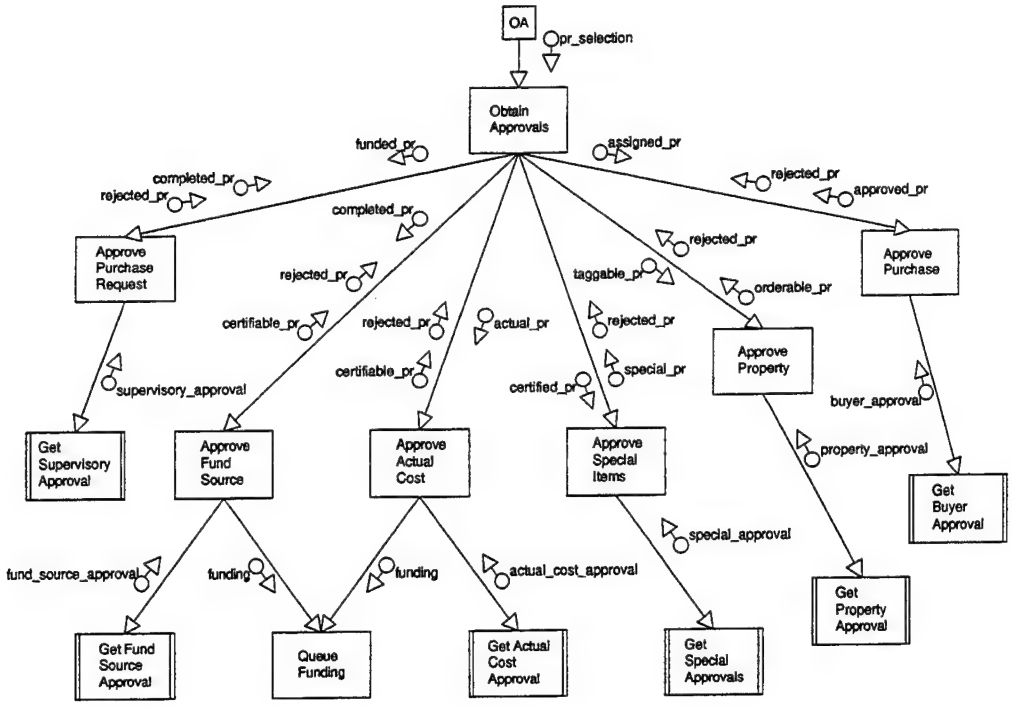

Figure 10. Obtain Approvals.

Uses: Approve Purchase Request, Approve Fund Source, Approve Actual Cost, Approve Special Items, Approve Property, Approve Purchase.

Returns: N/A.

Functional Details:

1. If user role is supervisor and pr_selection is a funded_pr, then Approve Purchase Request.
2. If user role is budget analyst and pr_selection is a completed_pr, then Approve Fund Source.
3. If user role is supervisor and pr_selection is a actual_pr, then Approve Actual Cost.
4. If user role is special approver and pr_selection is a certified_pr, then Approve Special Items.
5. If user role is property book and pr_selection is a taggable_pr, then Approve Property.

6. If user role is buyer and pr_selection is a assigned_pr, then Approve Purchase.
7. Store approved or rejected purchase request in the ACTIVE data store.

(2) **Module:** Approve Purchase Request.

Purpose: Obtain supervisory approval.

Uses: Get Supervisory Approval.

Returns: completed_pr, rejected_pr.

Functional Details:

1. Get Supervisory Approval from supervisor.
2. If approval is ""Yes," then determine budget analyst and set to author.
3. If approval is "No," then set author to requester.

(3) **Module:** Approve Fund Source.

Purpose: Obtain fund source approval from budget analyst.

Uses: Get Fund Source Approval, Queue Funding.

Returns: certifiable_pr, rejected_pr.

Functional Details:

1. Get Fund Source Approval from budget analyst.
2. If approval is ""Yes," then Queue Funding.
3. If approval is "No," then set author to requester.

(4) **Module:** Approve Actual Cost.

Purpose: Obtain actual cost approval from supervisor.

Uses: Get Actual Cost Approval, Queue Funding.

Returns: certifiable_pr, rejected_pr.

Functional Details:

1. Get Actual Cost Approval from supervisor.
2. If approval is "Yes," then Queue Funding.
3. If approval is "No," then set author to requester.

(5) **Module:** Queue Funding.

Purpose: Queue the funding information for certification by the SOMARDS interface.

Uses: N/A.

Returns: N/A.

Functional Details:

1. Trigger certification agent along with doc_ref_no.

(6) **Module:** Approve Special Items.

Purpose: Obtain approval for special items.

Uses: Get Special Approvals.

Returns: special_pr, rejected_pr.

Functional Details:

1. Get Special Approval from special approving official.
2. If approval is "Yes," then determine property book officer and set to author.
3. If approval is "No," then set author to requester.

(7) **Module:** Approve Property.

Purpose: Obtain property approval from the property book officer.

Uses: Get Property Approval.

Returns: orderable_pr, rejected_pr.

Functional Details:

1. Get Property Approval from supervisor.
2. If approval is "Yes," then determine contracting officer and set to author.
3. If approval is "No," then set author to requester.

(8) **Module:** Approve Purchase.

Purpose: Obtain purchase approval from the buyer.

Uses: Get Buyer Approval.

Returns: approved_pr, rejected_pr.

Functional Details:

1. Get Buyer Approval from buyer.
2. If approval is "Yes," then set to author to receiving.
3. If approval is "No," then return to requester.

5.5 Editing Purchase Request. Figure 11 shows the "Edit Purchase Request" process structure chart. This function allows users to edit various fields in the purchase request based on their role and the current state of the request.

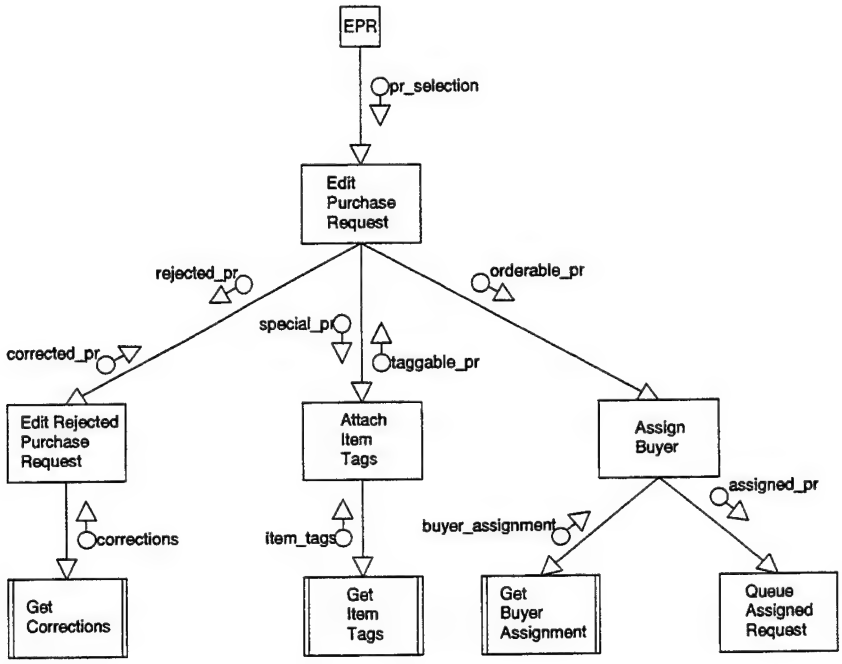

Figure 11. Edit Purchase Request.

Specifications for the five modules associated with Figure 11 are listed as follows.

(1) **Module:** Edit Purchase Request.

Purpose: Allow users to edit a purchase request based on their roles and the state of the purchase request.

Uses: Edit Rejected Purchase Request, Attach Item Tags, Assign Buyer.

Returns: N/A.

Functional Details:

1. If user is the requester and pr_selection is a rejected_pr, then Edit Rejected Purchase Request.
2. If user role is property and pr_selection is a special_pr, then Attach Item Tags.
3. If user role is contracting officer and pr_selection is a orderable_pr, then Assign Buyer.

(2) **Module:** Edit Rejected Purchase Request.

Purpose: Allow the requester to correct a purchase request that has been rejected.

Uses: Get Corrections.

Returns: corrected_pr.

Functional Details:

1. Get Corrections from the requester.
2. Clear all approvals and submit for supervisory approval.
3. Return corrected_pr to parent module.

(3) **Module:** Attach Item Tags.

Purpose: Allow the property book officer to attach Item Tags.

Uses: Get Item Tags.

Returns: taggable_pr.

Functional Details:

1. Get Item Tags from the property book officer until no more Item Tags.
2. Return taggable_pr to parent module.

(4) **Module:** Assign Buyer.

Purpose: Assign a buyer to a particular purchase request.

Uses: Get Buyer Assignment, Queue Assigned Request.

Returns: N/A.

Functional Details:

1. Get Buyer Assignment from contracting officer.
2. Queue Assigned Request for SAACONS upload.

(5) **Module:** Queue Assigned Request.

Purpose: Queue a purchase request for SAACONS upload.

Uses: N/A.

Returns: N/A.

Functional Details:

1. Trigger SAACONS upload agent with doc_ref_no.

5.6 Receive Shipment. Figure 12 shows the "Receive Shipment" process structure chart. This function allows receiving personnel to tag items and notify the requester of receipt.

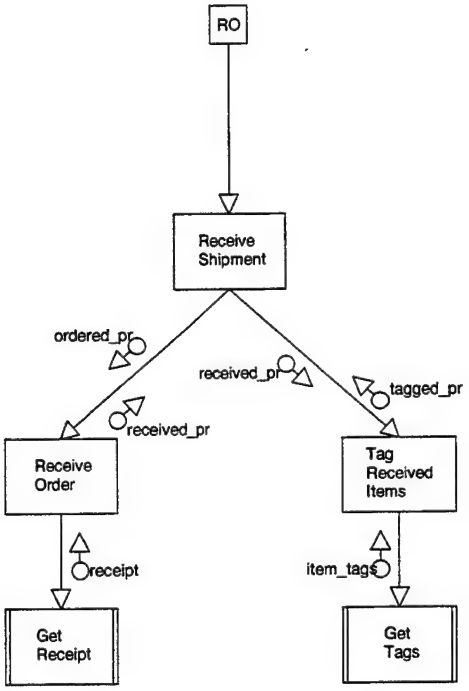

Figure 12. Receive Shipment.

Specifications for the three modules associated with Figure 12 are listed as follows.

(1) **Module:** Receive Shipment.

Purpose: Provide a means for Receiving to notify requesters of shipment receipt and to check taggable items.

Uses: Receive Order, Tag Received Items.

Returns: tagged_pr.

Functional Details:

1. Receive Order from Receiving.
2. Tag Received Items.
3. Set requester to author.

(2) **Module:** Receive Order.

Purpose: Allow Receiving to mark a request as received.

Uses: Get Receipt.

Returns: received_pr.

Functional Details:

1. Get Receipt from Receiving officer.
2. Return received_pr to parent module.

(3) **Module:** Tag Shipment Items.

Purpose: Allow Receiving to attach Item Tags to the taggable items.

Uses: Get Tags.

Returns: tagged_pr.

Functional Details:

1. Get Tags from Receiving until no more Tags.
2. Return tagged_pr to parent module.

5.7 Status Inquiry. Figure 13 shows the "Status Inquiry" process structure chart. This function allows users to check the current status of purchase requests that have been "touched" by them.

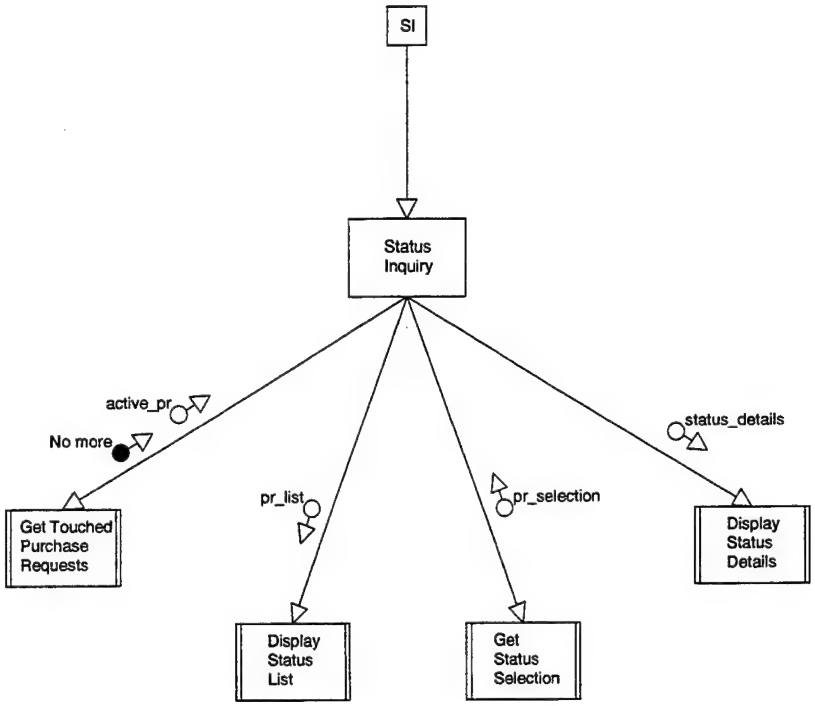

Figure 13. Status Inquiry.

Specifications for the one module associated with Figure 13 are listed as follows.

Module: Status Inquiry.

Purpose: Display the status of requests that have been "touched" by the user.

Uses: Get Touched Purchase Request, Display Status List, Get Status Selection, Display Status Details.

Returns: N/A.

Functional Details:

1. Get Touched Purchase Requests from ACTIVE data store until no more.
2. Display Status List of purchase requests to user.

3. Get Status Selection from user.

4. Display Status Details for pr_selection.

5.8 Generate Report. Figure 14 shows the "Generate Report" process structure chart. This function allows users of BuyIt to generate summary reports.

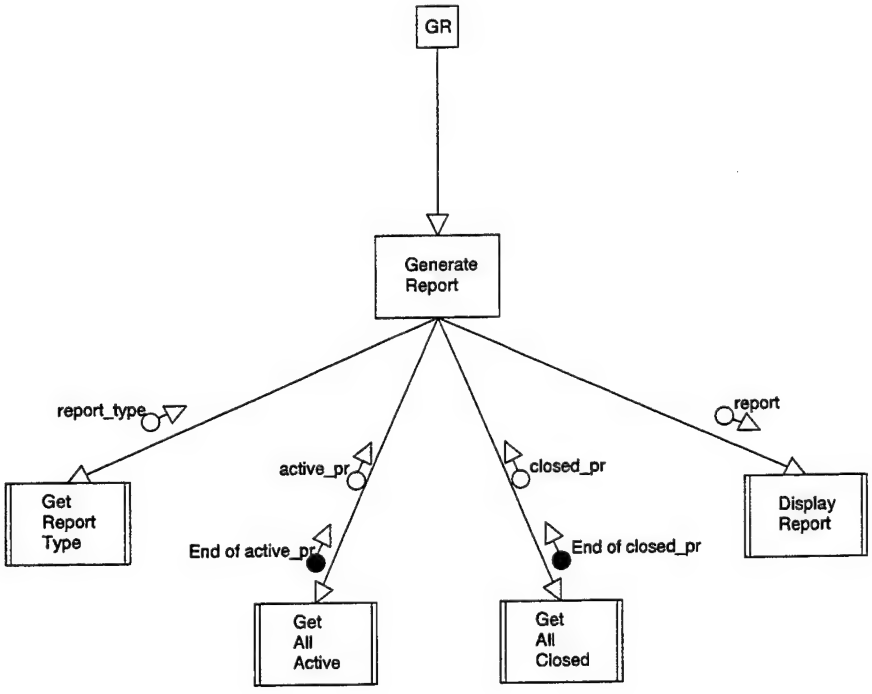

Figure 14. Generate Report.

Specifications for the one module associated with Figure 14 are listed as follows.

Module: Generate Report.

Purpose: Generate various reports that summarize the performance of the small purchase process.

Uses: Get Report Type, Get All Active, Get All Closed, Display Report.

Returns: N/A.

Functional Details:

1. Get Report Type from user.
2. Get All Active purchase request until no more.
3. Get All Closed purchase requests until no more.
4. Calculate summary data depending on report type.
5. Display report to user.

5.9 Legacy Interfaces. Figure 15 shows the "Legacy Interfaces" process structure chart. This function provides an automated interface between BuyIt and the legacy systems SOMARDS and SAACONS.

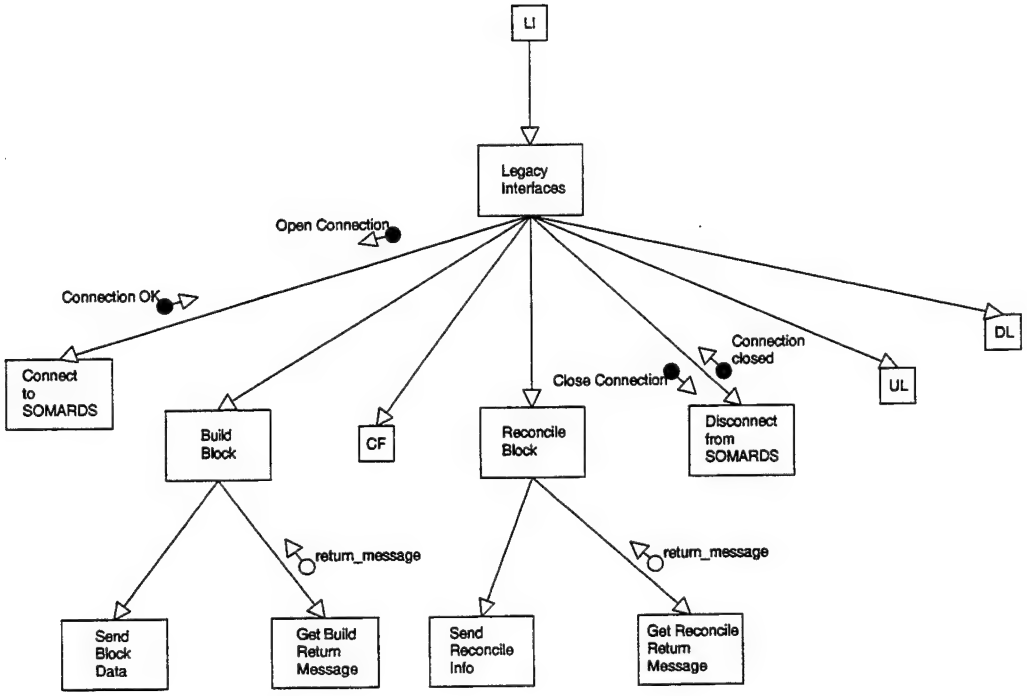

Figure 15. Legacy System Interfaces.

Specifications for the nine modules associated with Figure 15 are listed as follows.

(1) **Module:** Legacy Interfaces.

Purpose: Provide an automated interface between BuyIt and the legacy systems SOMARDS and SAACONS.

Uses: Connect to SOMARDS, Build Block, Certify Funds, Reconcile Block, Disconnect from SOMARDS, SAACONS Upload, SAACONS Download.

Returns: N/A.

Functional Details:

1. Connect to SOMARDS user interface.
2. Build Block in SOMARDS at start of day.
3. Certify Funds as they become available.
4. Reconcile Block at end of day.
5. Disconnect from SOMARDS interface.

(2) **Module:** Connect to SOMARDS.

Purpose: Provide an on-line interface to SOMARDS.

Uses: N/A.

Returns: connection_ok.

Functional Details:

1. Telnet to Rock Island IBM mainframe.
2. Proceed to SOMARDS login screen.
3. Login to SOMARDS.
4. If successful return connection_ok.

(3) **Module:** Build Block.

Purpose: Build a transaction block for BuyIt to use during the day.

Uses: Send Block Data, Get Build Return Message.

Returns: N/A.

Functional Details:

1. Send Block Data to SOMARDS connection.
2. Get Build Return Message.
3. Repeat steps 1 and 2 until return_message is successful.

(4) **Module:** Send Block Data.

Purpose: Sending transaction block data to SOMARDS for the morning build.

Uses: N/A.

Returns: N/A.

Functional Details:

1. Get today's date and format.
2. Send block to SOMARDS.

(5) **Module:** Get Build Return Message.

Purpose: Check the status of the build block success.

Uses: N/A.

Returns: N/A.

Functional Details:

1. Receive return_message from SOMARDS connection.
2. Return return_message to parent module.

(6) **Module:** Reconcile Block.

Purpose: Reconcile the SOMARDS block at the end of the day.

Uses: Send Reconcile Info, Get Reconcile Return Message.

Returns: N/A.

Functional Details:

1. Send Reconcile Info to SOMARDS connection to check balance.
2. Get Reconcile Return Message balance.
3. Check return_message balance against BuyIt total.
4. Send Reconcile Info to SOMARDS connection to adjust balance.
5. Get Reconcile Return Message.
6. Send Reconcile Info to SOMARDS connection to check balance.
7. Get Reconcile Return Message balance.
8. Check return_message balance against zero.

(7) **Module:** Send Reconcile Info.

Purpose: Send reconcile info to the SOMARDS connection.

Uses: N/A.

Returns: N/A.

Functional Details:

1. Send Reconcile Info to SOMARDS connection.

(8) **Module:** Get Reconcile Return Message.

Purpose: Receive the balance information from the SOMARDS interface.

Uses: N/A.

Returns: return_message.

Functional Details:

1. Receive return_message from SOMARDS connection.
2. Return return_message to parent module.

(9) **Module:** Disconnect from SOMARDS.

Purpose: To disconnect from the SOMARDS user interface.

Uses: N/A.

Returns: connection_closed.

Functional Details:

1. Send logout command to SOMARDS connection.
2. Wait for logout complete.
3. Return connection_closed to parent module.

5.10 Certify Funds. Figure 16 shows the "Certify Funds" process structure chart. This function provides an continuous certification process for BuyIt.

Specifications for the five modules associated with Figure 16 are listed as follows.

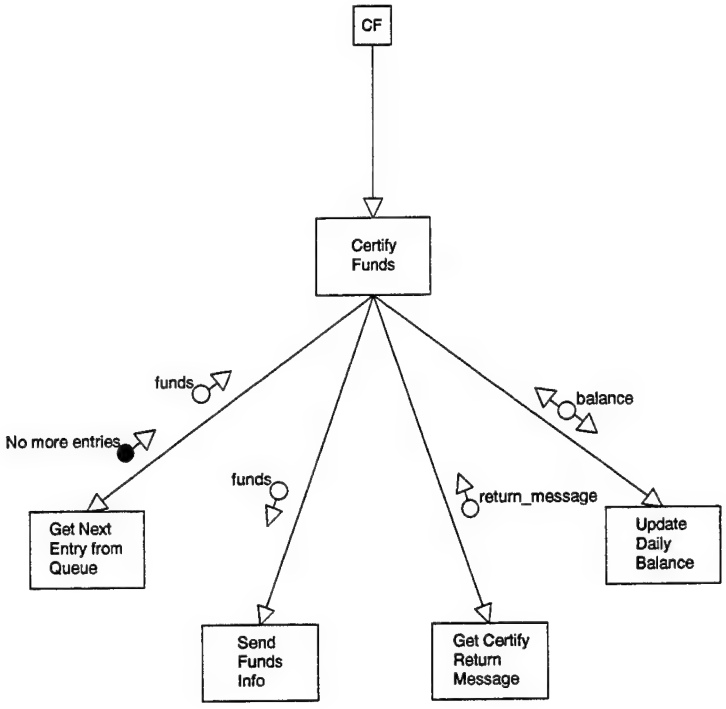

Figure 16. Certify Funds.

(1) **Module:** Certify Funds.

Purpose: Continuously certify funds.

Uses: Get Next Entry from Queue, Send Funds Info, Get Certify Return Message, Update Daily Balance.

Returns: N/A.

Functional Details:

1. Get Next Entry from Queue of certifiable purchase requests funding information.
2. Send Funds Info to SOMARDS connection.
3. Get Certify Return Message from SOMARDS connection.
4. Put the return_message in the certification return queue.
5. Update Daily Balance if successful.

(2) **Module:** Get Next Entry from Queue.

Purpose: Retrieve the next purchase request Funds Info.

Uses: N/A.

Returns: N/A.

Functional Details:

1. Scan the certifiable queue for the next entry.
2. Return the Funds Info. to the parent module.

(3) **Module:** Send Funds Info.

Purpose: Send the Funds Info. to the SOMARDS connection.

Uses: N/A.

Returns: N/A.

Functional Details:

1. Send the Funds Info. to the SOMARDS connection.

(4) **Module:** Get Certify Return Message.

Purpose: Retrieve the certification results.

Uses: N/A.

Returns: return_message.

Functional Details:

1. Receive return_message from SOMARDS connection.
2. Return return_message to parent module.

(5) **Module:** Update Daily Balance.

Purpose: To keep a running total of dollars certified.

Uses: N/A.

Returns: N/A.

Functional Details:

1. Calculate new balance using old balance plus the certification amount.
2. Store the new balance.

5.11 SAACONS Interface. Figure 17 shows the "SAACONS Upload" and "SAACONS Download" processes. These functions handle the flow of request and award data to and from SAACONS.

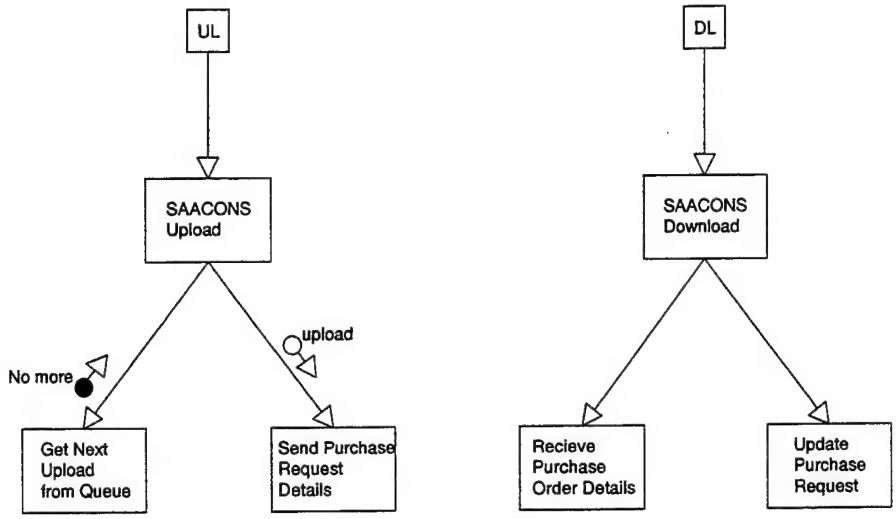

Figure 17. SAACONS Interface.

Specifications for the six modules associated with Figure 17 are listed as follows.

(1) **Module:** SAACONS Upload.

Purpose: Transfer purchase request details to SAACONS.

Uses: Get Next Upload from Queue, Send Purchase Request Details.

Returns: N/A.

Functional Details:

1. Get Next Upload from Queue until no more.
2. Format list for SAACONS.
3. Send Purchase Request Details to SAACONS.

(2) **Module:** Get Next Upload from Queue.

Purpose: Retrieve a purchase request details from the upload queue.

Uses: N/A.

Returns: N/A.

Functional Details:

1. Scan the upload queue for the next entry.
2. Return purchase request to parent module.

(3) **Module:** Send Purchase Request Details.

Purpose: Transfer purchase request upload data to SAACONS.

Uses: N/A.

Returns: N/A.

Functional Details:

1. Make FTP connection to SAACONS host.
2. Change to BuyIt to transfer directory.
3. Put upload file.
4. Close FTP connection.

(4) **Module:** SAACONS Download.

Purpose: Transfer award information from SAACONS and update purchase requests with award data.

Uses: Receive Purchase Order Details, Update Purchase Request.

Returns: N/A.

Functional Details:

1. Receive Purchase Order Details from SAACONS.
2. Format SAACONS download for update.
3. Update Purchase Request with award details using doc_ref_no.

(5) **Module:** Receive Purchase Order Details.

Purpose: Transfer award information from SAACONS.

Uses: N/A.

Returns: N/A.

Functional Details:

1. Make FTP connection to SAACONS host.
2. Change to BuyIt to transfer directory.
3. Get download file.
4. Close FTP connection.

(6) **Module:** Update Purchase Request.

Purpose: To update the BuyIt purchase request with award data.

Uses: N/A.

Returns: N/A.

Functional Details:

1. Load download file into temporary data store.
2. Scan temporary data store for purchase request.
3. Update BuyIt purchase request with award data.

5.12 Lotus Notes Agent Processes. Figure 18 shows the "Agent Processes" structure chart. This module handles the background process of the legacy system interface queues.

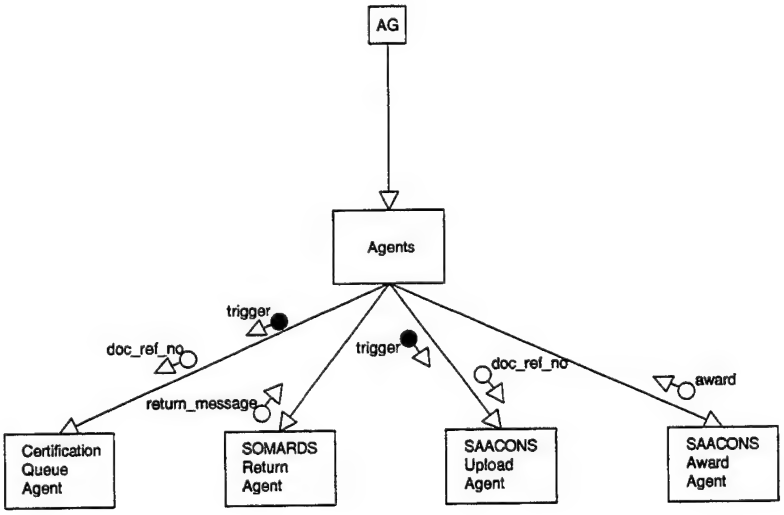

Figure 18. Agent Processes.

Specifications for the five modules associated with Figure 18 are listed as follows.

(1) **Module:** Agents.

Purpose: To start timed agents and trigger queue agents.

Uses: Certification Queue Agent, SOMARDS Return Agent, SAACONS Upload Agent, SAACONS Award Agent.

Returns: N/A.

Functional Details:

1. Start timers for SOMARDS Return Agent and SAACONS Award Agent
2. Wait for triggers and doc_ref_no from user interface.

(2) **Module:** Certification Queue Agent.

Purpose: To transfer funding information to the certification queue.

Uses: N/A.

Returns: N/A.

Functional Details:

1. Wait for user interface trigger.
2. Scan for purchase request using doc_ref_no.
3. Transfer funding info to certification queue.

(3) **Module:** SOMARDS Return Agent.

Purpose: To transfer SOMARDS returns from the results queue to the purchase request.

Uses: N/A.

Returns: N/A.

Functional Details:

1. Wait for timing trigger.
2. Scan SOMARDS results queue for entry.
3. If certification is "Yes," then mark purchase request as certified and set author to special approver.
4. If certification is "No," then set author to requester.

(4) **Module:** SAACONS Upload Agent.

Purpose: Transfer request information to the SAACONS upload queue.

Uses: N/A.

Returns: N/A.

Functional Details:

1. Wait for user interface trigger.
2. Scan for purchase request using doc_ref_no.
3. Transfer purchase request details to upload queue.

(5) **Module:** SAACONS Award Agent.

Purpose: Transfer award information from the SAACONS award queue to the purchase request.

Uses: N/A.

Returns: N/A.

Functional Details:

1. Wait for timing trigger.
2. Scan SAACONS award queue for entry.
3. Update purchase request with award data.

INTENTIONALLY LEFT BLANK.

6. References

1. Business Process Reengineering Project. "To-Be Model: Small Purchase." U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, July 1995.

2. Enterprise Systems Division. "BuyIt Software Development Plan." SPAS-SDP-97-100, U.S. Army Research Laboratory, Aberdeen Proving Ground, MD.

3. Enterprise Systems Division. "BuyIt Software Requirements Analysis." SPAS-SRA-97-110, U.S. Army Research Laboratory, Aberdeen Proving Ground, MD.

4. Yourdon, E. *Modern Structured Analysis*. Englewood Cliffs, NJ: Yourdon Press, 1989.

INTENTIONALLY LEFT BLANK.

NO. OF COPIES ORGANIZATION

2 DEFENSE TECHNICAL INFORMATION CENTER
DTIC DDA
8725 JOHN J KINGMAN RD
STE 0944
FT BELVOIR VA 22060-6218

1 HQDA
DAMO FDQ
D SCHMIDT
400 ARMY PENTAGON
WASHINGTON DC 20310-0460

1 OSD
OUSD(A&T)/ODDDR&E(R)
R J TREW
THE PENTAGON
WASHINGTON DC 20301-7100

1 DPTY CG FOR RDE HQ
US ARMY MATERIEL CMD
AMCRD
MG CALDWELL
5001 EISENHOWER AVE
ALEXANDRIA VA 22333-0001

1 INST FOR ADVNCD TCHNLGY
THE UNIV OF TEXAS AT AUSTIN
PO BOX 202797
AUSTIN TX 78720-2797

1 DARPA
B KASPAR
3701 N FAIRFAX DR
ARLINGTON VA 22203-1714

1 NAVAL SURFACE WARFARE CTR
CODE B07 J PENNELLA
17320 DAHLGREN RD
BLDG 1470 RM 1101
DAHLGREN VA 22448-5100

1 US MILITARY ACADEMY
MATH SCI CTR OF EXCELLENCE
DEPT OF MATHEMATICAL SCI
MAJ M D PHILLIPS
THAYER HALL
WEST POINT NY 10996-1786

NO. OF COPIES ORGANIZATION

1 DIRECTOR
US ARMY RESEARCH LAB
AMSRL DD
J J ROCCHIO
2800 POWDER MILL RD
ADELPHI MD 20783-1145

1 DIRECTOR
US ARMY RESEARCH LAB
AMSRL CS AS (RECORDS MGMT)
2800 POWDER MILL RD
ADELPHI MD 20783-1145

3 DIRECTOR
US ARMY RESEARCH LAB
AMSRL CI LL
2800 POWDER MILL RD
ADELPHI MD 20783-1145

ABERDEEN PROVING GROUND

4 DIR USARL
AMSRL CI LP (305)

NO. OF
COPIES ORGANIZATION

7 DIRECTOR
US ARMY RESEARCH LAB
AMSRL IS
J GRILLS
AMSRL IS ES
R ROSEN
AMSRL CS
COL K LOGAN
AMSRL CI E
W JERNIGAN
W MCCOY
B SCHALLHORN
AMSRL SE S
A SINDORIS
2800 POWDER MILL RD
ADELPHI MD 20783-1145

1 DIRECTOR
US ARMY RESEARCH LAB
AMSRL EA
P STAY
WSMR NM 88002-5501

ABERDEEN PROVING GROUND

10 DIR USARL
AMSRL CI C NIETUBICZ
AMSRL CI E D ULERY (6 CPS)
AMSRL HR SF B AMREIN
AMSRL WM IM R MCGEE
AMSRL CS AP D ORE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	July 1999	Final, 1 Jan - 31 Jan 97

4. TITLE AND SUBTITLE
BuyIt Detailed Design Report: A C-BASS Component

5. FUNDING NUMBERS
AC9UA4EC

6. AUTHOR(S)
Brian R. Schallhorn

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
AMSRL-CI-E
Aberdeen Proving Ground, MD 21005-5067

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-MR-451

9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES)

10.SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This document is the third in a series of reports resulting form a structured engineering process for the BuyIt Software Project. As part of the Corporate Business Application Software System (C-BASS) suite of work flow applications, BuyIt automates small purchase orders for the U.S. Army Research Laboratory (ARL). Three major sections of this report give background for the BuyIt Project, explain the structured design and methodology, and define the design environment. A fourth segment presents a detailed design for BuyIt using the formalisms of structure charts for defining operations and module specifications for defining objects.

14. SUBJECT TERMS
BuyIt, software structured design, software engineering, C-BASS

15. NUMBER OF PAGES
53

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	

INTENTIONALLY LEFT BLANK.

USER EVALUATION SHEET/CHANGE OF ADDRESS

This Laboratory undertakes a continuing effort to improve the quality of the reports it publishes. Your comments/answers to the items/questions below will aid us in our efforts.

1. ARL Report Number/Author ARL-MR-451 (Schallhorn) Date of Report July 1999

2. Date Report Received _____

3. Does this report satisfy a need? (Comment on purpose, related project, or other area of interest for which the report will be used.) _____

4. Specifically, how is the report being used? (Information source, design data, procedure, source of ideas, etc.) _____

5. Has the information in this report led to any quantitative savings as far as man-hours or dollars saved, operating costs avoided, or efficiencies achieved, etc? If so, please elaborate. _____

6. General Comments. What do you think should be changed to improve future reports? (Indicate changes to organization, technical content, format, etc.) _____

CURRENT ADDRESS

_____
Organization

_____
Name E-mail Name

_____
Street or P.O. Box No.

_____
City, State, Zip Code

7. If indicating a Change of Address or Address Correction, please provide the Current or Correct address above and the Old or Incorrect address below.

OLD ADDRESS

_____
Organization

_____
Name

_____
Street or P.O. Box No.

_____
City, State, Zip Code

(Remove this sheet, fold as indicated, tape closed, and mail.)
(DO NOT STAPLE)